**ESTRUTURAS DE DADOS 1 2022/1**

**Gabriel Ramiro Mesquita**

*Lista 03 – Complexidade*

*Respostas*

1. 1. N + M | O(n)
   2. N² | O(n²)
   3. (n/2) \* (log2(n)) | O(n log2(n))
2. Termo Dominante = T.D
   1. T.D: 0.001n5 | O(n5)
   2. T.D: 50nlog10(n) | O(nlog10(n))
   3. T.D: 2.5n1.75 | O(n1.75)
   4. T.D: n 2log2(n) | O(n 2log2(n))
   5. T.D: nlog2(n) | O(nlog2(n))
   6. T.D: 3log8(n) | O(log8(n)))
   7. T.D: 0.001n² | O(n²)
   8. T.D: 100n² | O(n²)
   9. T.D: 0.5n1.25 | O(n1.25)
   10. T.D: n(log2(n)² | O(n(log2(n)²)
   11. T.D: n³ | O(n³)
   12. T.D: log2(log2(n)) | O(log2(log2(n)))
3. 1. i) complexidade n, pois no pior dos casos todo o vetor vai ter que ser percorrido para concluir-se que o elemento desejado não está nele.
   2. ii) complexidade n, pelo mesmo motivo do algoritmo anterior.
   3. iii) complexidade log2(n), pois o algoritmo faz uma busca pelo valor desejado do tipo binary search. No pior dos casos o algoritmo vai dividir por 2 o tamanho total do vetor(n) y vezes, onde y = log2(n).
4. **Código inteiro em anexo.**

Alterei a classe pilha criando um novo atributo int pares. Em cada método push eu inseri uma condicional testando (int) elemento % 2 == 0, dependendo do resultado o próprio método puch muda ao tributio pares, e no final criei um método contaPares que retorna o valor da atribuído a variável pares. **Dessa forma os métodos funcionam com uma complexidade O(1).**

public Pilha(int tamanho)

{

n = tamanho;

vetor = new char[tamanho];

topo = -1;

pares = 0;

}

...

public char pop()

{

char c = '\0';

if (!this.vazia()){

c = vetor[topo];

if ((int)c %2 == 0){

pares--;

}

}

....

public boolean push(char elemento)

{

if (!this.cheia())

{

vetor[++topo] = elemento;

if ((int)elemento %2 == 0){

pares++;

}

return true;

…

public int contaPares(){

return pares;

}

1. **Complexidade assintótica O (n²)**

public void selectionSort(int vetor[], int n) {

int i,j;

for (j = 0; j < n-1; j++) { O(n)

int iMin = j; O (1)

for (i = j+1; i < n; i++) { O(n)

if (vetor[i] < vetor[iMin]) { O(1)

iMin = i; O(1)

}

}

if(iMin != j) { O(1)

int temp = vetor[j]; O(1)

vetor[j] = vetor[iMin]; O(1)

vetor[iMin] = temp; O(1)

}

}

}